Vol.2014-OS-128 No.5
2014/3/6

仮想マシンと物理マシンの一元管理を可能にする仮想 AMT

大薗 弘記[1,a)]　光来 健一[1]

概要：組織の管理者が管理しなければならない PC の数は膨大になってきている．管理者の負担を軽減するために，最近の PC には Active Management Technology（AMT）が搭載されるようになってきている．AMT を用いることで，管理者は PC をリモートから一元的に管理することができるようになる．しかし，近年，サーバ上で仮想マシン（VM）を動作させてその画面だけを PC に表示するという仮想デスクトップが普及してきている．これにより，組織内には PC と仮想デスクトップが混在しているが，AMT では PC の管理を行うことしかできないため，VM の管理は別に行う必要があった．本論文では，物理マシンと VM を一元的に管理できるようにするために VM に対して仮想的な AMT を提供する**仮想** *AMT*（*vAMT*）を提案する．vAMT は PC を管理する AMT と同様に，CIM と Web サービス，VNC のインタフェースを用いて VM を管理することを可能にする．AMT と vAMT を用いることで，PC と VM の違いを意識することなく，既存の AMT 用管理ツールを用いて一元的な管理を行うことができるようになる．vAMT を用いた実験を行い，AMT 対応の既存の管理ツールによって VM の管理が行えることを確認した．

1. はじめに

企業など組織の IT 化の進展に伴い，組織で使用される PC の数は膨大になってきている．組織の IT 部門の管理者はこれらすべての PC を管理しなければならない．管理の内容は，ヘルプデスクサポートやパッチ配布，ソフトウェアのアップグレード，ウィルス対策など多岐にわたる．例えば，離れた部署のユーザからの障害報告への対応を行う場合，問題になっているシステムが管理者の目の前にないために対応が難しいことがある．その場合，管理者は障害の発生した PC の設置場所まで赴き，問題箇所を特定してから，障害の原因によっては交換の必要な部品を取りに戻ることになる．OS が動作していて，リモートコントロール用のエージェントがその上で正常に動作している場合は，リモートでの障害対応もかなり容易ではある．しかし，OS そのものが起動しない場合や BIOS 設定を変更する必要がある場合，あるいはハードウェアに問題がある場合などは，ソフトウェアベースの管理だけでは対応しきれない．

そのような問題から，最近の PC には Active Management Technology（AMT）が搭載されるようになってきている．AMT はインテル社の vPro プロセッサに標準搭載されている機能で，マザーボード上に CPU とは別の組み込みプロセッサを備えることで，PC の電源が切れている状態からでもリモート操作や電源操作などが可能になる技術である．AMT を用いることで，管理者はリモートから PC を一元的に管理することができるようになる．AMT により PC をハードウェアレベルで管理することができるため，リモートで PC の OS 起動前の画面表示を確認したり，BIOS 設定画面をリモートで操作したりすることができ，管理者の負担は大きく軽減される．

しかし，近年，サーバの仮想マシン（VM）上でシステムを動作させ，その画面だけを PC 上で表示する仮想デスクトップが普及してきている．これにより，組織内は物理マシンである PC と VM が混在する環境になっている．このような環境において，AMT では物理マシンの管理を行うことしかできないため，VM の管理は別に行う必要がある．

そこで，本論文では物理マシンと VM を一元的に管理できるようにするために，VM 用の仮想的な AMT である**仮想** *AMT*（*vAMT*）を提案する．vAMT は物理マシンを管理する AMT と同様に，Common Information Model (CIM) [4] と Web サービス，VNC のインタフェースを用いて VM を管理することを可能にする．CIM を用いる場合，リモートの管理ツールは通信プロトコルに WS-Management [2] を用いて vAMT から情報を取得したり，管理を実行したりする．Web サービスを用いる場合，管理ツールは通信プロトコルに SOAP [3] を用いて vAMT にアクセスする．AMT と vAMT を用いることで，物理マシンと VM の違いを意識することなく，既存の AMT 用管理ツールを用いて一元的な管理を行うことができるようになる．

[1] 九州工業大学
Kyushu Institute of Technology, Iizuka, Fukuoka 820-8502, Japan
[a)] oozono@ksl.ci.kyutech.ac.jp

以下，2 章では AMT の基本的な機能や VM と PC の混在環境における問題について述べる．3 章では vAMT を提案し，4 章では vAMT が用いるインターフェースについて述べる．5 章では実装について説明し，6 章では vAMT を用いて行った実験について述べる．7 章では関連研究について述べ，8 章で本論文をまとめる．

2. 従来の PC 管理

2.1 AMT

AMT はインテル社が提供する vPro の管理機能の核となる技術であり，PC をハードウェアレベルで管理することができる．AMT の基本的な機能としては，検出，障害回復，保護がある．AMT はシステムの起動時に System Management BIOS（SMBIOS） [1] テーブルからハードウェアとソフトウェアに関する情報を取得し，それを独自のフラッシュメモリに格納している．これにより，PC の電源がオフの状態でもリモートから情報の取得を可能にする．

障害回復機能としては，PC に接続されているキーボードやマウス，PC の電源をリモートから操作することができる．AMT はハードウェアレベルで管理を行えるため，管理エージェントが起動していない状態でも自由に電源を操作できる．OS が起動しない場合は，管理者側にあるハードウェア診断ツールの入った起動ディスクイメージを使って対象の PC を起動し，問題のある箇所を特定することができる．

保護機能としては，OS 上で動く管理エージェントの動作を監視することができる．管理エージェントは定期的に AMT に対してハートビートを送っているので，ハートビートが停止した場合に AMT が管理者にアラートを送信したり，NIC の制御を行うことでネットワークへの接続を制限したりすることができる．

2.2 仮想デスクトップの普及

近年，仮想デスクトップと呼ばれる利用形態が普及してきている．仮想デスクトップはサーバ上で VM を動作させて，ネットワークを通じて VM に接続してその画面だけを PC に表示する．仮想デスクトップを使用することで，システムをサーバで集中管理することができ，ソフトウェアの追加や更新，修正などのメンテナンスが容易となる．仮想デスクトップの普及はまだ過渡期であることや，ネットワークが繋がらない環境で使用されるマシンは仮想デスクトップに置き換えることができないなどの問題により，現在，組織内では PC と仮想デスクトップが混在している．

そのため，組織内の全てのマシンの管理を行う場合，物理マシンである PC と仮想デスクトップである VM の両方を管理する必要がある．しかし，AMT は物理マシンの管理を行うための技術であるため，VM の管理には別の管理ツールを用いる必要がある．そのため，管理者は少なくとも 2 種類の管理ツールを使い分けなければならず，管理が煩雑になる．

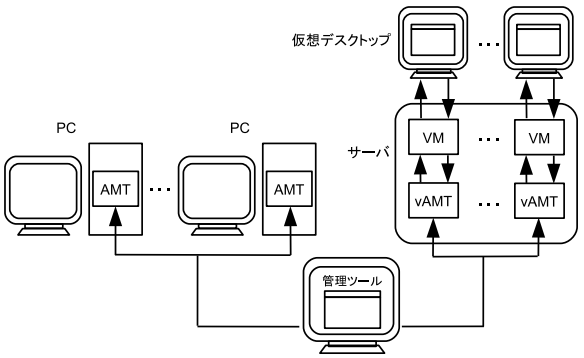

図 1 システムの全体構成

3. vAMT

本論文では，VM 用の仮想的な AMT である仮想 AMT（vAMT）を提案する．vAMT は PC を管理する AMT と同様に，CIM [4] と Web サービス，VNC のインタフェースを用いて VM を管理することを可能にする．リモートの管理ツールは CIM と Web サービスのリクエストを送ることで vAMT にアクセスを行うことができる．通信プロトコルにはそれぞれ WS-Management [2] と SOAP [3] が用いられる．また，VNC を用いて vAMT 経由で VM の帯域外リモート管理を行うことができる．AMT と vAMT を用いることで，物理マシンと VM の違いを意識することなく，既存の AMT 用管理ツールを用いて一元的な管理を行うことができるようになる．AMT と vAMT による管理システムの全体構成を図 1 に示す．

vAMT は図 2 のように構成される．管理ツールから WS-Management でリクエストが送られてきた場合，WS-Management サーバがリクエストを CIM のリクエストに変換して CIM オブジェクトマネージャ（CIMOM）に送る．CIMOM は変換された CIM のリクエストを適切な CIM プロバイダに送る．AMT の各機能ごとに CIM プロバイダを用意し，VM の仮想ハードウェアにアクセスして VM の情報を取得したり管理を実行したりする．一方，SOAP でリクエストが送られてきた場合は，SOAP サーバによって適切な Web サービスが呼び出される．Web サービスは CIM プロバイダと同様に VM の情報取得や操作を行う．

リモートの管理ツールが AMT にアクセスする時には，AMT が搭載された PC の IP アドレスを指定して接続を行う．同様に，vAMT にアクセスする時も，管理対象の VM の IP アドレスを指定して接続を行うことができる．vAMT は VM へのパケットを受信すると，16992 番ポートへのアクセスであれば Web サーバに振り分ける．Web サーバではリクエスト先の URL を用いて WS-Management サーバ

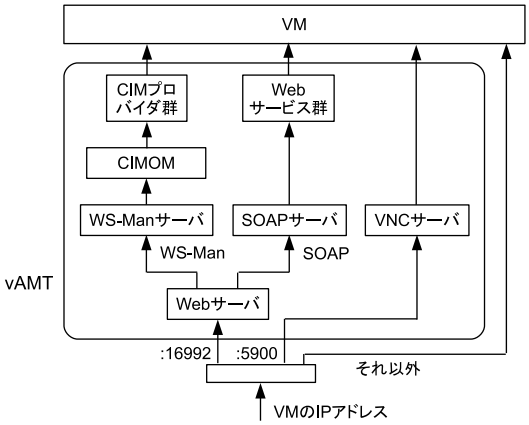

図 2 vAMT の構成

または SOAP サーバへの振り分けを行う．5900 番ポートへのアクセスであれば VNC サーバに振り分け，それ以外のポートへのアクセスは VM に振り分ける．

4. vAMT のインタフェース

4.1 CIM インタフェース

リモートの管理ツールが WS-Management によるアクセスを行う場合，vAMT から情報を取得したり，管理を実行したりするために，CIM を AMT 用に拡張したインタフェースを用いる．CIM は異なる管理対象オブジェクトの複雑な相互依存や関連性をたどったり，表現したりすることを容易にするためにオブジェクト指向の階層的アーキテクチャになっている．CIM を定義するための要素として，クラス，プロパティ，メソッド，修飾子，参照，関連がある．

クラス 特定の種類のオブジェクトに共通するプロパティやメソッドを定義する．

プロパティ クラスの特徴を表現するための値で，名前，データ型，値を持つ．

メソッド メソッドが定義されているクラスのインスタンスに対して操作を行うために呼び出される．

修飾子 クラス，メソッド，メソッドの引数，プロパティ，参照，関連に関する追加情報を提供する．

参照 他のインスタンスへのポインタであることを示す特別なデータ型である．

関連 2 つ以上の参照を含むクラスの一種であり，異なるクラスのインスタンスの関係を表す．

CIM のクラスを実際に定義するためには Managed Object Format（MOF）言語が用いられる．MOF は構文仕様のために BNF 記法を拡張したもので，CIM の仕様で定められている．MOF で記述された CIM クラスの例を図 3 に示す．この例では CIM_Processor，CIM_Chip，CIM_Realizes という 3 つのクラスが定義されている．ただし，説明を簡単にするために，実際の定義とは少し異なる．

CIM_Processor はデバイスの論理的な情報を扱うための

```
class CIM_Processor : CIM_LogicalDevice {
  [Key] uint32 Number;
  uint32 Enable([IN] boolean Enabled);
};

class CIM_Chip : CIM_PhysicalElement {
  [Key] string Tag;
};

class CIM_Realizes {
  [Key] CIM_PhysicalElement REF Antecedent;
  [Key] CIM_LogicalDevice REF Dependent;
};
```

図 3 MOF で記述された CIM クラスの例

CIM クラス CIM_LogicalDevice を継承したクラスで，CPU 番号を示す Number というプロパティと，CPU の有効化・無効化を行うための Enable というメソッドを持っている．Number に付いている [Key] という修飾子はそのプロパティがインスタンスを識別するのに用いられることを表しており，キープロパティと呼ばれる．また，Enable メソッドの引数である Enabled に付いている [IN] という修飾子はその引数が入力引数であることを示す．

CIM_Chip はデバイスを物理要素として見た時の情報を扱うための CIM クラス CIM_PhysicalElement を継承したクラスである．CIM_Realizes は CIM_PhysicalElement と CIM_LogicalDevice の参照型をプロパティに持つ関連クラスである．データ型の後の REF は参照型であることを表している．これは各デバイスの論理情報と物理情報を対応づけるためのクラスである．

CIM のクラスやインスタンスを操作するために，CIM オペレーションが用いられる．例えば，インスタンスの取得方法には EnumerateInstances と GetInstance の 2 種類がある．EnumerateInstances では CIM クラスのすべてのインスタンスを取得するのに対して，GetInstance ではキープロパティの値を指定することで 1 つのインスタンスのみを取得する．また，引数を指定して CIM クラスのメソッドを呼び出し，返り値を受け取ることもできる．

4.2 Web サービスインタフェース

管理ツールが SOAP によるアクセスを行う場合，AMT 用の Web サービスをインタフェースとして用いる．Web サービスを記述するには Web Services Description Language（WSDL）が使用される．WSDL は XML ベースの言語仕様であり，Web サービスがどのような機能を持つのか，それを利用するためにはどのような要求をすればいいのかなどを記述する方法が定義されている．WSDL 文書を構成する主な要素として，ポートタイプ，オペレーション，メッセージ，タイプがある．

ポートタイプ 関連するオペレーションをまとめて定義する．

```
<types>
  <element name="GetVersionResp"> …
    <element name="Version" type="string"/> ...
  </element>
</types>

<message name="GetVersionOut">
  <part name="parameters" element="GetVersionResp">
</message>

<portType name="SecurityAdminPortType">
  <operation name="GetVersion">
    <input message="GetVersionIn"/>
    <output message="GetVersionOut"/>
  </operation>
</portType>
```

図 4 WSDL で記述された Web サービスの例

オペレーション 入力メッセージ（引数）と出力メッセージ（返り値）を指定して操作（メソッド）を定義する．

メッセージ 型を指定してデータを定義する．

タイプ 型を定義する．

WSDL で記述された Web サービスの例を図 4 に示す．この例はセキュリティに関する管理を行う SecurityAdminPortType というポートタイプに AMT のバーションを取得するオペレーションである GetVersion を定義している．GetVersion オペレーションは入力メッセージとして GetVersionIn を，出力メッセージとして GetVersionOut を持っている．GetVersionOut メッセージは GetVersionResp という型を持ち，この型は Version という名前の文字列からなる．GetVersionIn メッセージの定義は省略しているが，GetVersion という型を持つ．

4.3 VNC インタフェース

管理ツールがリモートから VM の画面を確認したりキーボードやマウスを操作したりすることができるようにするために，VNC をインタフェースとして用いる．vAMT 経由で VNC 接続することで，VM 内の状態に依存しない帯域外リモート管理を行うことができる．AMT を経由した VNC 接続にはデフォルトポートを使用する方法とリダイレクションポートを使用する方法の 2 種類がある．デフォルトポートを使用すると，通常の VNC 接続を行うことができ，リダイレクションポートを使用すると，AMT 独自のプロトコルを用いて TLS によるセキュリティの高い接続を行える．

5. 実装

vAMT の実装は図 5 のように行った．CIMOM には OpenPegasus [5] を用いた．OpenPegasus で vAMT へのリクエストを処理できるようにするには 3 つの修正を行う必要があった．1 つ目は，CIM_で始まる CIM クラスだけでなく，AMT_や IPS_で始まる CIM クラスも同様に扱えるようにすることである．これらは AMT で独自に定義されている CIM クラスである．2 つ目は，CIM のリクエストで名前空間が指定されていないときにデフォルトの名前空間の “root/cimv2” を使うようにすることである．AMT の管理ツールの中には名前空間を指定しないものがある．3 つ目は，リクエストの参照先のアドレス指定が “default” のときに CIM サーバの IP アドレスと名前空間を指すようにすることである．

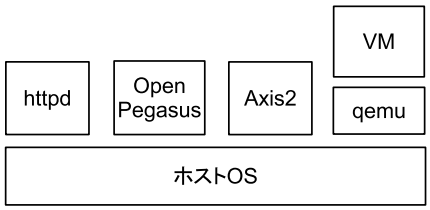

図 5 vAMT の実装

SOAP サーバには Apache Axis2 [7] を用いた．Axis2 は SOAP サーバを提供するだけでなく，Web サービスのための様々なツールも含んでいる．Web サーバには Apache HTTP サーバを用いた．

VNC サーバには Linux KVM の qemu-kvm に含まれる実装を用いた．qemu-kvm は VM を動かすための仮想化ソフトウェアであり，VM を動かしている間だけ存在する．

5.1 CIM プロバイダ

VM に対してアクセスを行う CIM プロバイダを作成するために CIMPLE [6] というツールを利用した．CIMPLE を用いることで，CIM クラスが定義されている MOF から CIM プロバイダの雛形を生成することができる．vAMT 用の CIM プロバイダを作成するにあたって，インテルが提供している MOF ファイルを用いた．

CIMPLE は CIM クラスに対応する C++のクラスと CIM プロバイダの雛形となるクラスを生成する．前者のクラスは CIM クラスのプロパティに対応するメンバを持つ．後者のクラスにはメンバ関数として enum_instances や get_instance などが定義されている．例えば，enum_instances 関数は CIM オペレーションの EnumerateInstances が実行される時に呼び出される．

VM の情報を取得したり操作を行ったりするには libvirt [8] というライブラリを使用した．libvirt では様々な仮想化ソフトウェア上で動作している VM を統一的に扱うことを可能にしている．libvirt を用いて VM にアクセスすることで，異なる仮想化ソフトウェアに対して 1 つの CIM プロバイダで対応できる．

図 3 の CIM_Processor のための CIM プロバイダに enum_instances 関数を記述する例を図 6 に示す．この例では CIM_Processor の create メソッドを用いて CPU の数だけインスタンスを作成し，各インスタンスに CPU 番号

```
Enum_Instances_Status
CIM_Processor_Provider::enum_instances(
  const CIM_Processor* model,
  Enum_Handler<CIM_Processor>* handler)
{
  for (i = 0; i < nCPUs; i++) {
    CIM_Processor *cpu = CIM_Processor::create();
    cpu->Number.set(i);
    handler->handle(cpu);
  }
  return ENUM_INSTANCES_OK;
}
```

図 6 インスタンスを列挙する例

```
Get_Instance_Status
CIM_Processor_Provider::get_instance(
  const CIM_Processor* model,
  CIM_Processor*& instance)
{
  int val = model->Number.value;
  if (val >= 0 && val < nCPUs){
    instance->Number.set(val);
    return GET_INSTANCE_OK;
  }
  return GET_INSTANCE_NOT_FOUND;
}
```

図 7 特定のインスタンスを返す例

を設定している．初期化を終えたインスタンスはハンドラに登録することで要求元に送られる．

get_instance 関数にはその CIM クラス中の 1 つのインスタンスを返すように記述する．get_instance の記述例を図 7 に示す．要求されたインスタンスの持つプロパティの値が引数 model に格納されており，一致するインスタンスが存在する場合にはそのインスタンスのプロパティ値を引数 instance に格納して要求元に返す．この例では，要求された CPU 番号を持つ CPU が存在すれば対応するインスタンスの情報を返している．enum_instances や get_instance によって CIM クラスが持つインスタンスを返すために用いられる CIM プロバイダをインスタンスプロバイダと呼ぶ．

図 8 は CIM_Processor の Enable メソッドの記述例である．引数 self で対象となるインスタンスが指定され，引数 Enabled でこのメソッドの引数が渡される．この関数の中で CPU の有効化・無効化後に VM に割り当てられる仮想 CPU の数を再計算し，libvirt の関数 virtDomainSetVcpus を用いて実際の割り当てを変更する．引数 return_value に返り値を設定することでメソッドの実行結果を返す．

図 3 の CIM_Realizes の CIM プロバイダに enum_instances 関数を記述する例を図 9 に示す．まず，関連づけを行う CIM_Chip と CIM_Processor のインスタンスをそれぞれ作成し，キープロパティに値を代入して初期化を行う．次に，CIM_Realizes のインスタンスを作成し，CIM_Chip と CIM_Processor のインスタンスをメンバの Antecedent と Dependent に代入する．この時，それぞれを

```
Invoke_Method_Status
CIM_Processor_Provider::Enable(
  const CIM_Processor* self,
  const Property<boolean>& Enabled,
  Property<uint32>& return_value)
{
  nCPUs = ...
  virDomainSetVcpus(nCPUs);
  return INVOKE_METHOD_OK;
}
```

図 8 メソッド呼び出しの例

```
Enum_Instances_Status
CIM_Realizes_Provider::enum_instances(
  const CIM_Realizes* model,
  Enum_Instances_Handler<CIM_Realizes>* handler)
{
  CIM_Chip* chip = CIM_Chip::create(true);
  chip->Tag.set("CPU 0");

  CIM_Processor* cpu = CIM_Processor::create(true);
  cpu->Number.set(0);

  CIM_Realizes* link = CIM_Realizes::create(true);
  link->Antecedent = cast<CIM_PhysicalElement*>(chip);
  link->Dependent = cast<CIM_LogicalDevice*>(cpu);
  handler->handle(link);
  return ENUM_INSTANCES_OK;
}
```

図 9 関連づけのためのインスタンスを返す例

CIM_PhysicalElement と CIM_LogicalDevice にキャストしている．このように，関連づけを行うために用いられる CIM プロバイダを関連プロバイダと呼ぶ．

現在のところ，262 個の vAMT 用の CIM クラスのうち，よく使われる 39 個について CIM プロバイダが実装できている．実現できている機能の一部を以下に示す．

- バージョン情報の取得
  管理ツールは CIM_SoftwareIdentity クラスを用いて AMT のバージョンを取得している．そこで，リクエストされた InstanceID プロパティの値が “AMT” の場合に vAMT のバージョン番号を MajorVersion プロパティなどに持つインスタンスを返すようにした．
- 電源状態の取得
  管理ツールは VM の電源状態を知るために CIM_AssociatedPowerManagementService クラスを用いている．そこで，libvirt の virDomainIsActive 関数を使って VM の電源状態を取得し，その結果を PowerState プロパティに設定したインスタンスを返すようにした．

5.2 Web サービス

VM の管理を行う Web サービスを作成するために Axis2 に含まれる WSDL2Java というツールを利用した．WSDL2Java によって，WSDL ファイルから Web サービス

```
public class SecurityAdminServiceSkeleton {
  public GetVersionResp getVersion(GetVersion version) {
    GetVersionResp resp = new GetVersionResp();
    resp.setVersion("8.0.0");
    return resp;
  }
}
```

図 10 Web サービスの記述例

の雛形を生成することができる．vAMT 用の Web サービスを作成するにあたって，インテルが提供している WSDL ファイルを用いた．

WSDL2Java によって WSDL から Web サービスのオペレーションの雛形とメッセージに対応するクラスが Java で生成される．生成された雛形の各メソッドに具体的な処理を記述する．CIM プロバイダと同様に，VM の情報を取得したり操作を行ったりするには libvirt を使用する．しかし，libvirt が直接サポートしている言語は C と C++であるため，libvirt-java [9] というバインディングツールを利用して Java から libvirt を呼び出せるようにした．作成した Web サービスは Axis2 アーカイブファイルとしてまとめた状態でデプロイすることで利用可能になる．

図 4 の WSDL から生成した SecurityAdminWeb サービスの雛形に GetVersion オペレーションを記述する例を図 10 に示す．SecurityAdminServiceSkeleton というクラスのメソッドとして，オペレーション名に対応する getVersion が生成される．このメソッドの引数には入力メッセージの型である GetVersion が指定され，返り値には出力メッセージの型である GetVersionResp が指定されている．この例では GetVersionResp の Version パラメータに vAMT のバージョン情報を格納している．

現在のところ，267 個の vAMT 用の Web サービスのオペレーションのうち，よく使われる 20 個が実装できている．実現できている機能の一部を以下に示す．

- 電源の操作
  管理ツールは VM の電源操作を行うために RemoteControlService の RemoteControl オペレーションを用いる．RemoteControlService は電源状態の管理や電源操作を行うための Web サービスである．getCommand メソッドを用いてどのような電源操作を要求されているかを調べ，要求に応じて virDomainShutdown などの適切な libvirt 関数を呼び出して VM の電源状態を切り替える．

5.3 停止状態の VM の管理

vAMT は停止状態の VM に対しても管理を行うことができる．VM は PC と異なり，停止状態の時には実体がない．そのため，停止状態の VM から情報を取得したり VNC 接続を行ったりすることはできない．そこで，VM の情報を取得したり操作を行ったりする際に libvirt を用いることで，VM のコンフィグから情報を取得する．VNC 接続に関しては，VM に接続できない時はダミーの VNC サーバに接続して，AMT と同様に黒い画面を返すようにすることで AMT と同じ挙動にすることができる．

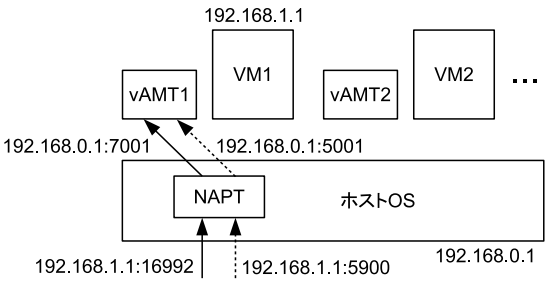

図 11 NAPT によるリクエストの振り分け

5.4 vAMT へのリクエストの振り分け

図 11 のようにホスト OS で静的 NAPT の設定を行い，vAMT へのアクセスを転送する．VM の IP アドレスに対する 16992 番ポート宛てのパケットは，ホスト OS の IP アドレスと vAMT が用いるポート番号に変換する．VNC 接続についても，VM の 5900 番ポートへのアクセスはホスト OS の vAMT のポート番号に変換する．これにより，管理ツールは AMT と同じアドレス，ポートの指定方法で vAMT にアクセスすることができる．

vAMT と管理対象の VM を同じ IP アドレスでアクセスできるようにするために，図 11 のように NAPT を用いて vAMT へのアクセスを振り分ける．vAMT はホスト OS 上で動作するいくつかのサーバ群で構成されているため，ホスト OS の IP アドレスが用いられる．一方，VM にはホスト OS とは異なる IP アドレスが割り当てられている．AMT は WS-Management や SOAP のリクエストを 16992 番ポートで待つため，VM の IP アドレスに対する 16992 番ポートへのアクセスは，ホスト OS の IP アドレスと vAMT が用いるポート番号（図 11 の例では 7001 番ポート）に変換する．同様に，AMT の VNC サーバは 5900 番ポートを用いるため，5900 番ポートへのアクセスはホスト OS の IP アドレスと vAMT が用いるポート番号（図 11 の例では 5001 番ポート）に変換する．このように NAPT を用いることで，別の VM を管理する vAMT を同じホスト OS 上で動かしても，vAMT に異なるポート番号を使わせることができる．

6. 実験

Intel System Defense Utility という AMT 用の管理ツールを使用して，vAMT に対して管理を実行した．まず，この管理ツールを用いて管理対象の VM に対応する vAMT への接続を行った．この管理ツールは接続の際に 97 個の

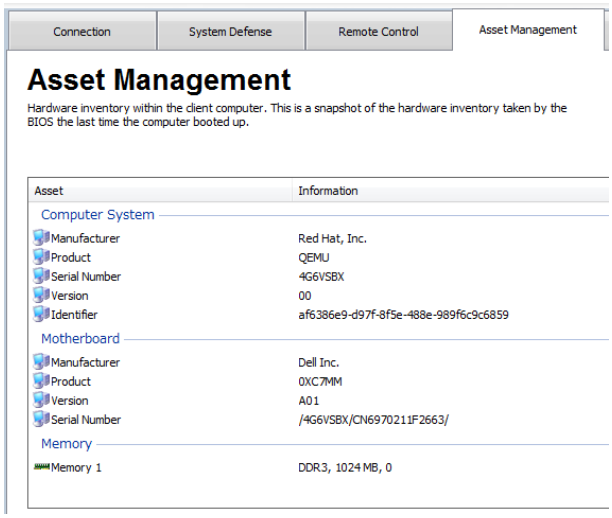

図 12 vAMT の資産管理画面

リクエストを送り，vAMT から管理に必要な様々な情報を取得する．これらのリクエストに対して適切に VM の情報を返すことで，接続に成功することを確認した．接続が完了すると，管理ツールの資産管理画面に管理対象マシンのハードウェア情報が表示された．vAMT の資産管理画面を図 12 に示す．コンピュータシステム情報の製造者と型番，識別子の 3 項目と，メモリ情報の容量の項目に VM の情報が表示された．コンピュータシステム情報のシリアル番号とバージョンの 2 項目，マザーボード情報の全項目，メモリ情報の種類に関しては，VM が動作しているホスト OS の情報が表示された．

次に，vAMT に対して VM の電源操作を実行した．AMT の場合と同様に CIM プロバイダと Web サービスが 1 つずつ使用されて，ツールの画面上に表示される電源状態が変化することを確認した．その後で，VM が動作しているホスト OS 上で，virsh という VM を管理するための CUI のツールを使用して電源状態を取得した．その結果，要求した通りに電源状態が切り替わっていることを確認した．

さらに，vAMT を経由した VM へのリモート接続を行った．デフォルトポートを使用した VNC 接続を行う場合は，CIM プロバイダや Web サービスは使用されない．したがって，NAPT の設定を行うだけで，図 13 のように VM へのリモート接続ができることを確認した．

最後に，AMT と vAMT に対して，AMT の SDK で提供されている AssetDisplay という管理コマンドを使用してシステム情報を取得するのにかかる時間を測定した．各 10 回ずつ測定を行い，平均を求めた結果を図 14 に示す．この結果より，vAMT の方が AMT よりも短い時間でコマンドを処理できていることが分かった．送られたリクエストの数はどちらも 12 個であったため，vAMT を搭載している PC 本体の CPU に比べて AMT のハードウェア性能が低いことが原因であると考えられる．

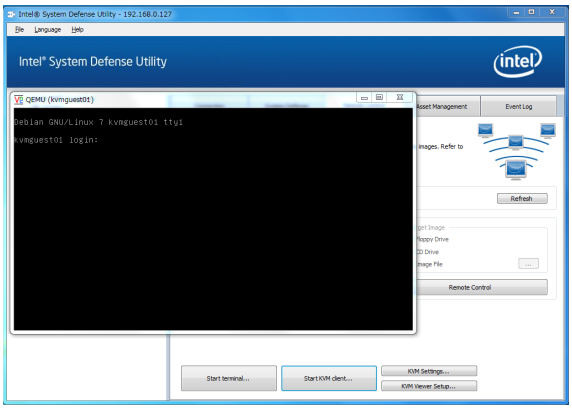

図 13 VNC による VM への帯域外リモート接続

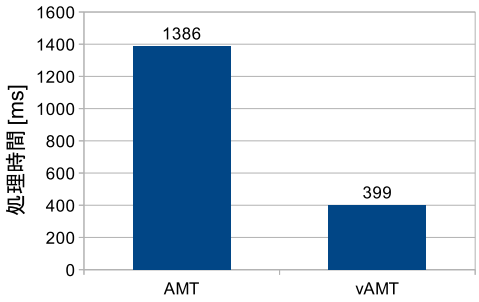

図 14 情報取得コマンドの実行時間

7. 関連研究

QND Plus [10] に代表されるソフトウェアベースの管理ツールでは，管理エージェントをインストールすることで PC と VM を一元的に管理することができる．ただし，管理エージェントの停止時や管理される PC の電源がオフの時には管理を行うことができないという問題がある．QND Plus を含めいくつかの管理ツールは AMT にも対応しているが，VM に対して AMT を用いた管理を行うことはできない．

Virt-manager [11] は libvirt を用いて実装された VM の管理ツールである．そのため，Xen や KVM，QEMU などの異なる仮想化ソフトウェアの VM を一括して管理することができる．Virt-manager は VM を外から管理するため，VM 内で管理エージェントを動作させる必要がない．しかし，Virt-manager では PC の管理を行うことはできない．

IPMI [12] は CPU や OS に依存することなくハードウェアを管理するためのサーバ向けの管理インタフェースである．AMT と同様に，サーバの温度や電源，ファンの状態などを監視できる．また，システムがダウンしている場合でもサーバシステムの状態をチェックしたり，管理機能を持つ別のサーバを接続してサーバを監視したりする機能を持っている．IPMI はサーバ向けであり，PC には搭載されていないため，PC と VM の一元管理には用いることがで

きない．

VM の管理を行えるようにするために，仮想化に対応した CIM [13] も定義されている．この CIM を用いることで物理マシンとその上で動作する VM を 1 つの管理ツールで管理することができる．しかし．仮想化対応の CIM は AMT の規格には含まれていない．また，物理マシンと VM を意識せずに扱うこともできないため，管理ツールでの対応が必要となる．それに対して，本研究では vAMT を用いることにより既存の管理ツールを利用することができる．

8. まとめ

本論文では，VM を管理するための仮想的な AMT である vAMT を提案した．AMT と同様のインタフェースを vAMT にも持たせることで，既存の管理ツールを用いて PC と VM を一元的に管理することができる．実際に既存の管理ツールを用いて PC と VM を意識せずに管理できることを確認した．

現在のところ，AMT を用いた管理に必要な CIM プロバイダと Web サービスの一部しか実装できていない．既存の管理ツールの持つあらゆる機能を利用するには，これらを全て実装する必要がある．また，AMT では管理エージェントの監視を行う保護機能のために，OS 側から AMT に対するアクセスを必要としている．そこで vAMT についても，VM 内の OS から使われるインタフェースを提供する必要がある．

参考文献

[1] DMTF: *System Management BIOS Reference Specification Version 2.7.1*, (2011).
[2] DMTF: *Web Services for Management Specification Version 1.1.1*, (2012).
[3] World Wide Web Consortium: *SOAP Version 1.2*, (2007).
[4] DMTF: *Common Information Model Infrastructure Version 2.7.0*, (2012).
[5] The Open Group: *OpenPegasus*, available from ⟨http://www.openpegasus.org/⟩.
[6] K. Schopmeyer and M. Brasher: *CIMPLE*, available from ⟨http://simplewbem.org/⟩.
[7] Apache Software Foundation: *Apache Axis2*, available from ⟨http://axis.apache.org/⟩.
[8] Red Hat: *libvirt*, available from ⟨http://libvirt.org/⟩.
[9] Red Hat: *libvirt-java*, available from ⟨http://libvirt.org/⟩.
[10] QualitySoft: *QND Plus*, available from ⟨http://www.quality.co.jp/⟩.
[11] Red Hat: *Virt-manager*, available from ⟨http://virt-manager.org/⟩.
[12] Intel, Hewlett-Packard, NEC, and Dell: *Intelligent Platform Management Specification Second Generation v2.0*, (2004).
[13] DMTF: *CIM System Virtualization Model Version 1.0.0*, (2007).